2019

Numerically stable coded matrix computations via circulant and rotation matrix embeddings

Aditya Ramamoorthy
Iowa State University, adityar@iastate.edu

Li Tang
Iowa State University, litang@iastate.edu

Numerically stable coded matrix computations via circulant and rotation matrix embeddings

Abstract

Several recent works have used coding-theoretic ideas for mitigating the effect of stragglers in distributed matrix computations (matrix-vector and matrix-matrix multiplication) over the reals. In particular, a polynomial code based approach distributes matrix-matrix multiplication among n worker nodes by means of polynomial evaluations. This allows for an ``optimal" recovery threshold whereby the intended result can be decoded as long as at least (n−s) worker nodes complete their tasks; s is the number of stragglers that the scheme can handle. However, a major issue with these approaches is the high condition number of the corresponding Vandermonde-structured recovery matrices. This presents serious numerical precision issues when decoding the desired result.

It is well known that the condition number of real Vandermonde matrices grows exponentially in n. In contrast, the condition numbers of Vandermonde matrices with parameters on the unit circle are much better behaved. In this work we leverage the properties of circulant permutation matrices and rotation matrices to obtain coded computation schemes with significantly lower worst case condition numbers; these matrices have eigenvalues that lie on the unit circle. Our scheme is such that the associated recovery matrices have a condition number corresponding to Vandermonde matrices with parameters given by the eigenvalues of the corresponding circulant permutation and rotation matrices. We demonstrate an upper bound on the worst case condition number of these matrices which grows as ≈O(ns+6). In essence, we leverage the well-behaved conditioning of complex Vandermonde matrices with parameters on the unit circle, while still working with computation over the reals. Experimental results demonstrate that our proposed method has condition numbers that are several orders of magnitude better than prior work.

Disciplines

Computer Sciences | Electrical and Computer Engineering

Comments

This is a pre-print of the article Ramamoorthy, Aditya, and Li Tang. "Numerically stable coded matrix computations via circulant and rotation matrix embeddings." *arXiv preprint arXiv:1910.06515* (2019). Posted with permission.

Numerically stable coded matrix computations via circulant and rotation matrix embeddings

Aditya Ramamoorthy and Li Tang
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
{adityar,litang}@iastate.edu

Abstract

Several recent works have used coding-theoretic ideas for mitigating the effect of stragglers in distributed matrix computations (matrix-vector and matrix-matrix multiplication) over the reals. In particular, a polynomial code based approach distributes matrix-matrix multiplication among n worker nodes by means of polynomial evaluations. This allows for an "optimal" recovery threshold whereby the intended result can be decoded as long as at least $(n-s)$ worker nodes complete their tasks; s is the number of stragglers that the scheme can handle. However, a major issue with these approaches is the high condition number of the corresponding Vandermonde-structured recovery matrices. This presents serious numerical precision issues when decoding the desired result.

It is well known that the condition number of real Vandermonde matrices grows exponentially in n. In contrast, the condition numbers of Vandermonde matrices with parameters on the unit circle are much better behaved. In this work we leverage the properties of circulant permutation matrices and rotation matrices to obtain coded computation schemes with significantly lower worst case condition numbers; these matrices have eigenvalues that lie on the unit circle. Our scheme is such that the associated recovery matrices have a condition number corresponding to Vandermonde matrices with parameters given by the eigenvalues of the corresponding circulant permutation and rotation matrices. We demonstrate an upper bound on the worst case condition number of these matrices which grows as $\approx O(n^{s+6})$. In essence, we leverage the well-behaved conditioning of complex Vandermonde matrices with parameters on the unit circle, while still working with computation over the reals. Experimental results demonstrate that our proposed method has condition numbers that are several orders of magnitude better than prior work.

1 Introduction

Present day computing needs necessitate the usage of large computation clusters that regularly process huge amounts of data on a regular basis. In several of the relevant

application domains such as machine learning, datasets are often so large that they cannot even be stored in the disk of a single server. Thus, both storage and computational speed limitations require the computation to be spread over several worker nodes. Such large scale clusters also present attendant operational challenges. These clusters (which can be heterogeneous in nature) suffer from the problem of "stragglers", which are defined as slow nodes (node failures are an extreme form of a straggler). The overall speed of a computational job on these clusters is typically dominated by stragglers in the absence of a sophisticated assignment of tasks to the worker nodes.

In recent years, approaches based on coding theory (referred to as "coded computation") have been effectively used for straggler mitigation [1–9]. Coded computation offers significant benefits for specific classes of problems such as matrix computations. We illustrate this by means of a matrix-vector multiplication example [4]. Suppose that a master node wants to compute $\mathbf{A}^T\mathbf{x}$ where the matrix $\mathbf{A}$ is very large. It can block decompose $\mathbf{A}^T = [\mathbf{A}_0^T \ \mathbf{A}_1^T]$ and assign three worker nodes the tasks of determining $\mathbf{A}_0^T\mathbf{x}$, $\mathbf{A}_1^T\mathbf{x}$ and $(\mathbf{A}_0^T + \mathbf{A}_1^T)\mathbf{x}$ respectively. It is easy to see that even if one worker node fails, there is enough information for a master node to compute the final result. More generally, these methods allow the master node to recover $\mathbf{A}^T\mathbf{x}$ if any τ of the worker nodes complete their computation; τ is called the recovery threshold. However, this requires the master node to solve simple systems of equations. This approach can be generalized for matrix multiplication by using Reed-Solomon (RS) code like approaches [1–5]. The work of [1], poses the multiplication of two matrices in a form that is roughly equivalent to a Reed-Solomon code. In particular, each worker node's task (which is multiplying smaller submatrices) can be imagined as a coded symbol. As long as enough tasks are complete, the master node can recover the matrix product by polynomial interpolation.

While polynomials allow for an elegant way to "embed" distributed matrix computations into the structure of an erasure code, in practice they suffer from serious numerical precision issues. It is well-recognized that polynomial interpolation over the reals suffers from several numerical stability issues owing to the high condition numbers of the corresponding Vandermonde matrices [10]. This issue was first highlighted within the coded computation domain in [2] and has been discussed in some recent works [11–16] (we discuss these in more detail in the upcoming Section 2). Thus, a straightforward application of these ideas is impractical even for clusters with tens of worker nodes. For addressing these issues, several alternate coding techniques have been presented in the literature that have their relative benefits and drawbacks.

In this work, we demonstrate that matrices with significantly lower (worst-case) condition numbers can be obtained by the judicious usage of objects such as rotation matrices and circulant permutation matrices. This paper is organized as follows. Section 2 discusses the background, related work and summarizes our main contributions. Section 3 overviews certain structured matrices and their properties which turn out to be useful in our work. Sections 4 and 5 contain a description of our proposed techniques for distributed matrix-vector and distributed matrix-matrix multiplication respectively. Section 6 compares our

results with prior work.

2 Background, Related Work and Summary of Contributions

Consider a scenario where the master node has a large $t \times r$ matrix $\mathbf{A} \in \mathbb{R}^{t \times r}$ and either a $t \times 1$ vector $\mathbf{x} \in \mathbb{R}^{t \times 1}$ or a $t \times w$ matrix $\mathbf{B} \in \mathbb{R}^{t \times w}$. The master node wishes to compute $\mathbf{A}^T\mathbf{x}$ or $\mathbf{A}^T\mathbf{B}$ in a distributed manner over n worker nodes in the matrix-vector and matrix-matrix setting respectively. Towards this end, the master node partitions $\mathbf{A}$ (respectively $\mathbf{B}$) into Δ_A (respectively Δ_B) block columns. Each worker node is assigned $\ell_A \leq \Delta_A$ and $\ell_B \leq \Delta_B$ linearly encoded block columns of $\mathbf{A}_0, \ldots, \mathbf{A}_{\Delta_A - 1}$ and $\mathbf{B}_0, \ldots, \mathbf{B}_{\Delta_B - 1}$, so that the corresponding storage fractions are $\gamma_A = \ell_A/\Delta_A$ and $\gamma_B = \ell_B/\Delta_B$ respectively. These encoded block columns are denoted by the $\hat{}$ superscript.

Thus, in the matrix-vector case, the i-th worker is assigned $\hat{\mathbf{A}}_{i,j}$, $0 \leq j < \ell_A$ and the vector $\mathbf{x}$; it is responsible for computing $\hat{\mathbf{A}}_{i,j}^T\mathbf{x}$ for all $0 \leq j < \ell_A$. In the matrix-matrix case it is assigned both $\hat{\mathbf{A}}_{i,j}$, $0 \leq j < \ell_A$ and $\hat{\mathbf{B}}_{i,j}$, $0 \leq j < \ell_B$ and computes *all* pair-wise products $\hat{\mathbf{A}}_{i,k}^T\hat{\mathbf{B}}_{i,l}, k \in [\ell_A], l \in [\ell_B]$ where for a positive integer m, $[m]$ denotes the set $\{0, 1, \ldots, m-1\}$. Each worker node transmits the result of each of its computations to the master node as soon as it is complete. We say the worker node i has completed its job if all its computations have been transmitted to the master node.

Definition 1. *Computation Threshold.* We say that a given scheme has threshold τ if the master node can decode the intended result as long as *any* τ out of n worker nodes complete their jobs. In this case we say that the scheme is resilient to $s = n - \tau$ stragglers.

A significant amount of prior work [1, 5, 17, 18] has demonstrated interesting and elegant approaches based on embedding the distributed matrix computation into the structure of polynomials. In particular, [1] demonstrates that when $\ell_A = \ell_B = 1$ the optimal threshold is $\Delta_A\Delta_B$ and that polynomial based approaches (henceforth referred to as polynomial codes) achieve this threshold. We demonstrate this result by means of the following example.

Example 1. Consider the matrix-matrix multiplication scenario where $\Delta_A = \Delta_B = 2$ and $\ell_A = \ell_B = 1$. Consider the matrix polynomials

$$\mathbf{A}(z) = \mathbf{A}_0 + \mathbf{A}_1 z, \quad \text{and} \quad \mathbf{B}(z) = \mathbf{B}_0 + \mathbf{B}_1 z^2.$$

Suppose that the master node evaluates $\mathbf{A}(z)$ and $\mathbf{B}(z)$ at distinct real values $z_1, \ldots, z_n$. It sends $\hat{\mathbf{A}}_{i,0} = \mathbf{A}(z_i)$ and $\hat{\mathbf{B}}_{i,0} = \mathbf{B}(z_i)$ to the i-th worker node, which computes $\hat{\mathbf{A}}_{i,0}^T\hat{\mathbf{B}}_{i,0}$. We note here that

$$\mathbf{A}^T(z)\mathbf{B}(z) \;=\; (\mathbf{A}_0^T + \mathbf{A}_1^T z)(\mathbf{B}_0 + \mathbf{B}_1 z^2) \;=\; \mathbf{A}_0^T\mathbf{B}_0 + \mathbf{A}_1^T\mathbf{B}_0 z + \mathbf{A}_0^T\mathbf{B}_1 z^2 + \mathbf{A}_1^T\mathbf{B}_1 z^3.$$

Thus, as soon as *any* four out of the n worker nodes return the results of their computation, the master node can perform polynomial interpolation to recover the (k, l)-th entry of each

$\mathbf{A}_i^T\mathbf{B}_j$ for $0 \leq k < r/2$ and $0 \leq l < w/2$. Therefore, such a system is resilient to $n-4$ failures. It can be seen that the computational load on each worker node is 1/4-th of computation required for calculating $\mathbf{A}^T\mathbf{B}$.

Prior work has also considered other ways in which matrices $\mathbf{A}$ and $\mathbf{B}$ can be partitioned. For instance, they can be partitioned both along rows and columns. The work of [17, 18] has obtained threshold results in those cases as well. The so called Entangled Polynomial and Mat-Dot codes [17, 18], also use polynomial encodings followed by interpolation at the master node.

Vandermonde matrices play a key role when analyzing polynomial interpolation.

Definition 2. *Vandermonde Matrix.* A $m \times m$ Vandermonde matrix $\mathbf{V}$ with parameters $z_1, z_2, \ldots, z_m \in \mathbb{C}$ is such that

$$\mathbf{V}_{ij} = z_j^i, i \in [m], j \in [m].$$

If the z_i's are distinct, then $\mathbf{V}$ is nonsingular. In this work, we will also assume that the z_i's are non-zero.

We note here that in Example 1 above, polynomial interpolation at the master node corresponds to solving a 4×4 Vandermonde-system of equations. In general, it can be shown that the master node needs to solve a $\Delta_A\Delta_B \times \Delta_A\Delta_B$ Vandermonde system. The condition number of these matrices grows exponentially in $\Delta_A\Delta_B$. This is a significant drawback and even for systems with around $\Delta_A\Delta_B \approx 30$, the condition number is so large that the decoded results are essentially useless.

The main goal of our work is to consider alternate embeddings of distributed matrix computations that are significantly better behaved from a numerical precision perspective.

2.1 Related Work

The issue of numerical stability in the coded computation context has been considered in a few recent works [2, 11–16]. The work of [11, 13] presented strategies for distributed matrix-vector multiplication and demonstrated some schemes that empirically have better numerical performance than polynomial based schemes for some values of n and s. However, both these approaches work only for the matrix-vector problem. The work of [14] presents a random convolutional coding approach that applies for both the matrix-vector and the matrix-matrix multiplications problems. Their work demonstrates a computable upper bound on the worst case condition number of the decoding matrices by drawing on connections with the asymptotic analysis of large Toeplitz matrices. The recent preprint [16] presents constructions that are based on random linear coding ideas where the encoding coefficients are chosen at random from a continuous distribution. These exhibit better condition number properties.

The work most closely related to the our work is [15] considers an alternative approach for polynomial based schemes by working within the basis of orthogonal polynomials. They demonstrate an upper bound on the worst case condition number of the decoding matrices which grows as $O(n^{2s})$ where s is the number of stragglers that the scheme is resilient to. They also demonstrate experimentally that their performance is significantly better than the polynomial code approach. In contrast we demonstrate an upper bound that is $\approx O(n^{s+6})$. Furthermore, in Section 6 we show that in practice our worst case condition numbers are far better than [15].

2.2 Main contributions of our work

The work of [10] shows that unless all (or almost all) the parameters of the Vandermonde matrix lie on the unit circle, its condition number is badly behaved. However, most of these parameters are complex-valued (except ± 1), whereas our matrices $\mathbf{A}$ and $\mathbf{B}$ are real-valued. Using complex evaluation points in the polynomial code scheme, will increase the cost of computations approximately four times for matrix-matrix multiplication and around two times for matrix-vector multiplication. This is an unacceptable hit in computation time.

Our main finding in this paper is that we can work with real-valued embeddings that allow us to leverage the optimal threshold properties of Vandermonde matrices, while enjoying the low condition number of Vandermonde matrices with all parameters on the unit circle. Our specific contributions include the following.

- We demonstrate that rotation matrices and circulant permutation matrices of appropriate sizes can be used within the framework of polynomial codes. Specifically, circulant permutation matrices can be used for matrix-vector multiplication whereas rotation matrices can be used for both matrix-vector and matrix-matrix multiplication.

- Using these embeddings we show that the worst case condition number over all $\binom{n}{\tau}$ possible recovery matrices is upper bounded by $\approx O(n^{s+6})$. Furthermore, our experimental results indicate that the actual values are significantly smaller, i.e., the analysis yields pessimistic upper bounds.

We briefly overview some of the notation that we use in the paper. For a matrix $\mathbf{A}$, we will use $\mathbf{A}(i,j)$ to represent its (i,j)-th entry. We use MATLAB inspired notation at certain places. For instance $\text{diag}(a_1, a_2, \ldots, a_m)$ denotes a $m \times m$ matrix with a_i's on the diagonal and $\mathbf{A}(:,j)$ denotes the j-th column of matrix $\mathbf{A}$. The notation $\mathbf{A} \otimes \mathbf{B}$ denotes the Frolicker product of matrices $\mathbf{A}$ and $\mathbf{B}$.

3 Overview of structured matrices

Our schemes in this work will be defined by the encoding matrices used by the master node, which are such that the master node only needs to perform scalar multiplications and

additions. The computationally intensive tasks, i.e., matrix operations are performed by the worker nodes. To facilitate the description of our schemes, we need to define certain classes of matrices and discuss their relevant properties.

Definition 3. *Rotation matrix.* Let $\mathrm{i} = \sqrt{-1}$. The matrix 2×2 matrix $\mathbf{R}_\theta$ below is called a rotation matrix.

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \mathbf{Q}\Lambda\mathbf{Q}^*, \text{ where} \tag{1}$$

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathrm{i} & -\mathrm{i} \\ 1 & 1 \end{bmatrix}, \text{ and}$$

$$\Lambda = \begin{bmatrix} e^{\mathrm{i}\theta} & 0 \\ 0 & e^{-\mathrm{i}\theta} \end{bmatrix}. \tag{2}$$

Thus, the rotation matrix has the simple, yet useful property that it is a "real" matrix with complex eigenvalues $e^{\pm \mathrm{i}\theta}$ that lie on the unit circle, which can be specified with θ. We use this property extensively in the sequel.

A $m \times m$ permutation matrix is a binary matrix that has exactly a single "one" entry in each row and column and zeros elsewhere. A $m \times m$ circulant matrix is specified by its first row. The remaining rows are obtained by cyclicly shifting the first row with the shift index equal to the row index.

Definition 4. *Circulant Permutation Matrix.* A $m \times m$ matrix which is both a circulant and a permutation is called a circulant permutation matrix. Let $\mathbf{e}$ be a row vector of length m with $\mathbf{e} = [0\ 1\ 0\ \ldots\ 0]$ and $\mathbf{P}$ denote a circulant permutation with $\mathbf{e}$ as its first row. It can be shown that all $m \times m$ circulant permutations can be expressed as powers of $\mathbf{P}$, denoted by $\mathbf{P}^i, i = 0, \ldots, m-1$ where $\mathbf{P}^0 = \mathbf{I}_m$.

Let $\mathbf{W}$ denote the m-point DFT matrix, i.e., $\mathbf{W}_{i,j} = \frac{1}{\sqrt{m}}\omega_m^{ij}$ for $0 \leq i, j \leq m-1$ where $\omega_m = e^{-\mathrm{i}\frac{2\pi}{\mathrm{m}}}$ denotes the m-th root of unity. It is well known that $\mathbf{W}$ is unitary and diagonalizes $\mathbf{P}$ so that

$$\mathbf{P} = \mathbf{W}\mathrm{diag}(1, \omega_m, \omega_m^2, \ldots, \omega_m^{(m-1)})\mathbf{W}^*, \tag{3}$$

where the superscript $*$ denotes the complex conjugate operator.

Let $||\mathbf{M}||$ denote the maximum singular value of a matrix $\mathbf{M}$ of dimension $l \times l$.

Definition 5. *Condition number.* The condition number of a $l \times l$ matrix $\mathbf{M}$ is defined as $\kappa(\mathbf{M}) = ||\mathbf{M}||||\mathbf{M}^{-1}||$. It is infinite if the minimum singular value of $\mathbf{M}$ is zero.

Consider the system of equations $\mathbf{My} = \mathbf{z}$, where $\mathbf{z}$ is known and $\mathbf{y}$ is to be determined. If $\kappa(\mathbf{M}) \approx 10^b$, then the decoded result loses approximately b digits of precision [19]. In particular, matrices that are ill-conditioned lead to significant numerical problems when solving linear equations.

3.1 Condition Number of Vandermonde Matrices

Let $\mathbf{V}$ be a $m \times m$ Vandermonde matrix with parameters $s_0, s_1, \ldots s_{m-1}$. The following facts about $\kappa(\mathbf{V})$ follow from prior work [10].

- *Real Vandermonde matrices.* If $s_i \in \mathbb{R}, i \in [m]$, i.e., if $\mathbf{V}$ is a real Vandermonde matrix, then it is known that its condition number is exponential in m.
- *Complex Vandermonde matrices with parameters outside the unit circle.* Suppose that the s_i's are complex and let $s_+ = \max_{i=0}^{m-1} |s_i|$. If $s_+ > 1$ then $\kappa(\mathbf{V})$ is exponential in m. Furthermore, if $1/|s_i| \geq \nu > 1$ for at least $\ell \leq m$ of the m parameters, then $\kappa(\mathbf{V})$ is exponential in ℓ.

Based on the above facts, the only scenario where the condition number is somewhat well-behaved is if most or all of the parameters of $\mathbf{V}$ are complex and lie on the unit-circle. In the Appendix, we show the following result.

Theorem 1. Consider a $m \times m$ Vandermonde matrix $\mathbf{V}$ where $m < q$ (where q is odd) with distinct parameters $\{s_0, s_1, \ldots, s_{m-1}\} \subset \{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\}$. Then,

$$\kappa(\mathbf{V}) \leq O(q^{q-m+6}).$$

Thus, if $q - m$ is a constant, then $\kappa(\mathbf{V})$ grows only polynomially in q. In the subsequent discussion, we will leverage this fact extensively.

4 Distributed Matrix-Vector Multiplication

Recall from Example 1 that the basic idea in polynomial codes is to form matrix polynomials at the master node and evaluate them at distinct real values. Thus, each worker node gets a distinct evaluation and computes the product of its assigned submatrices.

At the top level, the overall idea in our work is to replace evaluations at real values by evaluations at suitable matrices. This allows us to work with real-valued computations while simultaneously leveraging the "eigenvalues" of the embedding (rotation or circulant permutation) matrices. As argued above, these matrices have their eigenvalues on the unit circle; this in turn provides the numerical stability advantage. In this section we discuss the matrix-vector multiplication problem. It can be handled by both rotation matrix and circulant permutation matrix embeddings.

4.1 Circulant Permutation Embedding

Let q be a *prime* number which is greater than or equal to n (number of worker nodes) and let τ be the desired threshold of the scheme. We partition $\mathbf{A}$ into $\Delta_A = \tau(q-1)$ block columns which are indexed as $\mathbf{A}_{i,j}, 0 \leq i < \tau, 0 \leq j < q-1$. The storage fraction is set

to $\gamma_A = \frac{q}{\tau(q-1)}$, so that $\ell_A = q$. Furthermore, the master node generates the following "precoded" matrices.

$$\mathbf{A}_{i,q-1} = -\sum_{j=0}^{q-2} \mathbf{A}_{i,j}, 0 \leq i < \tau. \tag{4}$$

The coded submatrices $\hat{\mathbf{A}}_{i,j}$ for $0 \leq i < n, 0 \leq j < q$ are generated by means of a $\tau q \times nq$ matrix $\mathbf{G}$ as follows.

$$\hat{\mathbf{A}}_{i,j} = \sum_{k\in[\tau],l\in[q]} \mathbf{G}(kq+l, iq+j)\mathbf{A}_{k,l}. \tag{5}$$

Let $\mathbf{I}$ denote the identity matrix and $\mathbf{P}$ be the $q \times q$ circulant permutation matrix introduced in Definition 4. Consider the following choice of matrix $\mathbf{G}$.

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{I} & \mathbf{P} & \mathbf{P}^2 & \cdots & \mathbf{P}^{n-1} \\ \mathbf{I} & \mathbf{P}^2 & \mathbf{P}^4 & \cdots & \mathbf{P}^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I} & \mathbf{P}^{\tau-1} & \mathbf{P}^{2(\tau-1)} & \cdots & \mathbf{P}^{(n-1)(\tau-1)} \end{bmatrix}. \tag{6}$$

The master node transmits $\hat{\mathbf{A}}_{i,j}$ for $j = 0, \ldots, q-1$ and the vector $\mathbf{x}$ to worker node i which sequentially computes the matrix-vector product $\hat{\mathbf{A}}_{i,j}^T\mathbf{x}$ for $j = 0, 1, \ \ldots \ , q-1$.

Remark 1. The $\hat{\mathbf{A}}_{i,j}$'s can simply be generated by additions since $\mathbf{G}$ is a binary matrix.

Theorem 2. The threshold for the circulant permutation based scheme specified above is τ. Furthermore, the worst case condition number of the recovery matrices is upper bounded by $O(q^{q-\tau+6})$.

Proof. Suppose that the workers indexed by $i_0, \ldots, i_{\tau-1}$ complete their tasks. The corresponding block columns of $\mathbf{G}$ can be extracted to form

$$\tilde{\mathbf{G}} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{P}^{i_0} & \mathbf{P}^{i_1} & \cdots & \mathbf{P}^{i_{\tau-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{i_0(\tau-1)} & \mathbf{P}^{i_1(\tau-1)} & \cdots & \mathbf{P}^{i_{\tau-1}(\tau-1)} \end{bmatrix}.$$

We note here that the decoder attempts to recover each entry of $\mathbf{A}_{i,j}^T\mathbf{x}$ from the results sent by the worker nodes. Thus, we can equivalently analyze the decoding by considering the system of equations as discussed below.

Let $\mathbf{m}, \mathbf{c} \in \mathbb{R}^{1\times \tau q}$ be row-vectors such that

$$\mathbf{m} = [\mathbf{m}_0, \cdots, \mathbf{m}_{\tau-1}] = [\mathbf{m}_{0,0}, \cdots, \mathbf{m}_{0,q-1}, \cdots, \mathbf{m}_{\tau-1,0}, \cdots, \mathbf{m}_{\tau-1,q-1}], \text{ and}$$
$$\mathbf{c} = [\mathbf{c}_{i_0}, \cdots, \mathbf{c}_{i_{\tau-1}}] = [\mathbf{c}_{i_0,0}, \cdots, \mathbf{c}_{i_0,q-1}, \cdots, \mathbf{c}_{i_{\tau-1},0}, \cdots, \mathbf{c}_{i_{\tau-1},q-1}],$$

where we note that not all variables in $\mathbf{m}$ are independent owing to (4). Let $\mathbf{W}$ represent the q-point DFT matrix. We let $\mathbf{m}^{\mathcal{F}}$ and $\mathbf{c}^{\mathcal{F}}$ denote the q-point "block-Fourier" transforms of these vectors, i.e,

$$\mathbf{m}^{\mathcal{F}} = \mathbf{m}\begin{bmatrix}\mathbf{W} & & \\ & \ddots & \\ & & \mathbf{W}\end{bmatrix} \text{ and}$$
$$\mathbf{c}^{\mathcal{F}} = \mathbf{c}\begin{bmatrix}\mathbf{W} & & \\ & \ddots & \\ & & \mathbf{W}\end{bmatrix}.$$

Let $\tilde{\mathbf{G}}_{k,l} = \mathbf{P}^{i_l k}$ denote the (k,l)-th block of $\tilde{\mathbf{G}}$. Using (3), we have

$$\tilde{\mathbf{G}}_{k,l} = \mathbf{W}\text{diag}(1, \omega_q^{i_l k}, \omega_q^{2i_l k}, \ldots, \omega_q^{(q-1)i_l k})\mathbf{W}^*.$$

Let $\tilde{\mathbf{G}}^{\mathcal{F}}_{k,l} = \text{diag}(1, \omega_q^{i_l k}, \omega_q^{2i_l k}, \ldots, \omega_q^{(q-1)i_l k})$, and $\tilde{\mathbf{G}}^{\mathcal{F}}$ represent the $\tau \times \tau$ block matrix with $\tilde{\mathbf{G}}^{\mathcal{F}}_{k,l}$ for $k, l = 0, \ldots, \tau-1$ as its blocks. Consider the system of equations

$$\mathbf{m}\tilde{\mathbf{G}} = \mathbf{c},$$

which can further be written as

$$\mathbf{m}\begin{bmatrix}\mathbf{W} & & \\ & \ddots & \\ & & \mathbf{W}\end{bmatrix}\begin{bmatrix}\mathbf{W}^* & & \\ & \ddots & \\ & & \mathbf{W}^*\end{bmatrix}\tilde{\mathbf{G}}\begin{bmatrix}\mathbf{W} & & \\ & \ddots & \\ & & \mathbf{W}\end{bmatrix} = \mathbf{c}\begin{bmatrix}\mathbf{W} & & \\ & \ddots & \\ & & \mathbf{W}\end{bmatrix},$$

$$\implies [\mathbf{m}_0^{\mathcal{F}}, \cdots, \mathbf{m}_{\tau-1}^{\mathcal{F}}]\tilde{\mathbf{G}}^{\mathcal{F}} = [\mathbf{c}_{i_0}^{\mathcal{F}}, \cdots, \mathbf{c}_{i_{\tau-1}}^{\mathcal{F}}].$$

Next, we note that as each block within $\tilde{\mathbf{G}}^{\mathcal{F}}$ has a diagonal structure, we can rewrite the system of equations in a block diagonal matrix upon applying an appropriate permutation (*cf.* Claim 2 in Appendix). Thus, we can rewrite it as

$$[\mathbf{m}_0^{\mathcal{F},\pi}, \cdots, \mathbf{m}_{q-1}^{\mathcal{F},\pi}]\tilde{\mathbf{G}}_d^{\mathcal{F}} = [\mathbf{c}_0^{\mathcal{F},\pi}, \cdots, \mathbf{c}_{q-1}^{\mathcal{F},\pi}], \tag{7}$$

where the permutation π is such that $\mathbf{m}_j^{\mathcal{F},\pi} = [\mathbf{m}_{0,j}^{\mathcal{F}}\ \mathbf{m}_{1,j}^{\mathcal{F}}\ \ldots\ \mathbf{m}_{\tau-1,j}^{\mathcal{F}}]$ and likewise $\mathbf{c}_j^{\mathcal{F},\pi} = [\mathbf{c}_{i_0,j}^{\mathcal{F}}\ \mathbf{c}_{i_1,j}^{\mathcal{F}}\ \ldots\ \mathbf{c}_{i_{\tau-1},j}^{\mathcal{F}}]$. Furthermore, $\tilde{\mathbf{G}}_d^{\mathcal{F}}$ is a block-diagonal matrix where each block is of

size $\tau \times \tau$. Now, according to (4), we have $\mathbf{m}_{i,0}^{\mathcal{F}} = \sum_{j=0}^{q-1} \mathbf{m}_{i,j} = 0$ for $i = 0, \ldots, \tau - 1$, which implies that $\mathbf{m}_0^{\mathcal{F},\pi}$ is a $1 \times \tau$ zero row-vector and thus $\mathbf{c}_0^{\mathcal{F},\pi}$ is too.

In what follows, we show that each of other diagonal blocks of $\tilde{\mathbf{G}}_d^{\mathcal{F}}$ is non-singular. This means that $[\mathbf{m}_0^{\mathcal{F}}, \cdots, \mathbf{m}_{\tau-1}^{\mathcal{F}}]$ and consequently $\mathbf{m}$ can be determined by solving the system of equations in (7). Towards this end, we note that the k-th diagonal block ($1 \leq k \leq q-1$) of $\tilde{\mathbf{G}}_d^{\mathcal{F}}$, denoted by $\tilde{\mathbf{G}}_d^{\mathcal{F}}[k]$ can be expressed as follows.

$$\tilde{\mathbf{G}}_d^{\mathcal{F}}[k] = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \omega_q^{i_0 k} & \omega_q^{i_1 k} & \cdots & \omega_q^{i_{\tau-1} k} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_q^{(\tau-1) i_0 k} & \omega_q^{(\tau-1) i_1 k} & \cdots & \omega_q^{(\tau-1) i_{\tau-1} k} \end{bmatrix}. \tag{8}$$

The above matrix is a *complex* Vandermonde matrix with parameters $\omega_q^{i_0 k}, \ldots, \omega_q^{i_{\tau-1} k}$. Thus, as long these parameters are distinct, $\tilde{\mathbf{G}}_d^{\mathcal{F}}[k]$ will be non-singular. Note that we need the property to hold for $k = 1, \ldots, q-1$. This condition can be expressed as

$$(i_\alpha - i_\beta) k \not\equiv 0 \pmod{q},$$

for $i_\alpha, i_\beta \in \{0, \ldots, n-1\}$ and $1 \leq k \leq q-1$. A necessary and sufficient condition for this to hold is that q is prime. An application of Theorem 1 shows that $\kappa(\tilde{\mathbf{G}}_d^{\mathcal{F}}[k]) \leq O(q^{q-\tau+6})$ for all k. As decoding $\mathbf{m}$ is equivalent to solving systems of equations specified by $\tilde{\mathbf{G}}_d^{\mathcal{F}}[k]$ for $1 \leq k \leq q-1$, the worst case condition number is at most $O(q^{q-\tau+6})$. □

The above proof suggests a natural decoding algorithm where the fast Fourier transform(FFT) plays a key role (see Algorithm 1).

Claim 1. The decoding complexity of recovering $\mathbf{A}^T\mathbf{x}$ is $O(r(\log q + \log^2 \tau))$.

Proof. Note that Algorithm 1 is applied for recovering the corresponding entries of $\mathbf{A}_{i,j}^T\mathbf{x}$ for $0 \leq i < \tau, 0 \leq j < q-1$ separately. There are $r/(\tau(q-1))$ such entries. The complexity of computing a N-point FFT is $O(N \log N)$ in terms of the required floating point operations (flops). Computing the permutation does not cost any flops and its complexity is negligible as compared to the other steps. Step 1 of Algorithm 1 therefore has complexity $O(\tau q \log q)$. In Step 2, we solve the degree $\tau - 1$ polynomial interpolation, $(q-1)$ times. This takes $O((q-1)\tau \log^2 \tau)$ [20]. Finally, Step 3, requires applying the inverse permutation and the inverse FFT; this requires $O(\tau q \log q)$ operations. Therefore, the overall complexity is given by

$$\begin{aligned} &\frac{r}{\tau(q-1)} \left(O(\tau q \log q) + O((q-1)\tau \log \tau^2) \right) \\ &\approx O(r(\log q + \log^2 \tau)). \end{aligned}$$

□

Algorithm 1 Decoding Algorithm of Distributed Matrix-vector Multiplication

1. procedure: Block Fourier Transform and Permute $\mathbf{c}$
for $j = 0$ **to** $\tau - 1$ **do**
Apply FFT to $\mathbf{c}_{i_j} = [\mathbf{c}_{i_j,0}, \cdots, \mathbf{c}_{i_j,q-1}]$ to obtain $\mathbf{c}^{\mathcal{F}}_{i_j} = [\mathbf{c}^{\mathcal{F}}_{i_j,0}, \cdots, \mathbf{c}^{\mathcal{F}}_{i_j,q-1}]$.
end for
Permute $\mathbf{c}^{\mathcal{F}} = [\mathbf{c}^{\mathcal{F}}_{i_0}, \cdots, \mathbf{c}^{\mathcal{F}}_{i_{\tau-1}}]$ by π to obtain $\mathbf{c}^{\mathcal{F},\pi} = [\mathbf{c}^{\mathcal{F},\pi}_0, \cdots, \mathbf{c}^{\mathcal{F},\pi}_{q-1}]$ where $\mathbf{c}^{\mathcal{F},\pi}_j = [\mathbf{c}^{\mathcal{F}}_{i_0,j}, \mathbf{c}^{\mathcal{F}}_{i_1,j}, \cdots, \mathbf{c}^{\mathcal{F}}_{i_{\tau-1},j}]$, for $j = 0, \ldots, q-1$.
end procedure
2. procedure: Decode $\mathbf{m}^{\mathcal{F},\pi}$ from $\mathbf{c}^{\mathcal{F},\pi}$
For $1 \leq i \leq q-1$, we decode $\mathbf{m}^{\mathcal{F},\pi}_i$ from $\mathbf{c}^{\mathcal{F},\pi}_i$ by polynomial interpolation or matrix inversion.
Set $\mathbf{m}^{\mathcal{F},\pi}_0 = [0, \cdots, 0]$.
end procedure
3. procedure: Inverse permute and block Fourier transform $\mathbf{m}^{\mathcal{F},\pi}$
Permute $\mathbf{m}^{\mathcal{F},\pi}$ by π^{-1} to obtain $\mathbf{m}^{\mathcal{F}} = [\mathbf{m}^{\mathcal{F}}_0, \cdots, \mathbf{m}^{\mathcal{F}}_{\tau-1}]$. Apply inverse FFT to each $\mathbf{m}^{\mathcal{F}}_i$ in $\mathbf{m}^{\mathcal{F}}$ to obtain $\mathbf{m} = [\mathbf{m}_0, \cdots, \mathbf{m}_{\tau-1}]$.
end procedure

4.2 Rotation Matrix Embedding

Let $\tilde{q}$ be an odd number such that $\tilde{q} \geq n$ and $\tau < n$ be the desired threshold. We partition $\mathbf{A}$ into $\Delta = 2\tau$ block columns indexed as $\mathbf{A}_{i,j}, 0 \leq i < \tau, j = 0, 1$. The coded submatrices are generated by means of the $2\tau \times 2n$ matrix $\mathbf{G}'$ as follows.

$$\hat{\mathbf{A}}_{i,j} = \sum_{k \in [\tau], l \in \{0,1\}} \mathbf{G}'(2k + l, 2i + j)\mathbf{A}_{k,l}. \tag{9}$$

Let $\theta = 2\pi/\tilde{q}$. Then,

$$\mathbf{G}' = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{I} & \mathbf{R}_\theta & \mathbf{R}^2_\theta & \cdots & \mathbf{R}^{n-1}_\theta \\ \mathbf{I} & \mathbf{R}^2_\theta & \mathbf{R}^4_\theta & \cdots & \mathbf{R}^{2(n-1)}_\theta \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I} & \mathbf{R}^{\tau-1}_\theta & \mathbf{R}^{2(\tau-1)}_\theta & \cdots & \mathbf{R}^{(n-1)(\tau-1)}_\theta \end{bmatrix}. \tag{10}$$

As before the master node transmits the $\hat{\mathbf{A}}_{i,0}, \hat{\mathbf{A}}_{i,1}$ and the vector $\mathbf{x}$ to worker node i which computes $\hat{\mathbf{A}}^T_{i,j}\mathbf{x}$.

Theorem 3. The threshold for the rotation matrix based scheme specified above is τ. Furthermore the worst case condition number of the recovery matrices is upper bounded by $O(\tilde{q}^{\tilde{q}-\tau+6})$.

Proof. The proof of this result is quite similar to the proof of Theorem 2. Suppose that workers indexed by $i_0, \ldots, i_{\tau-1}$ complete their tasks. We extract the corresponding block columns of $\mathbf{G}'$ to obtain

$$\tilde{\mathbf{G}}' = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{R}_\theta^{i_0} & \mathbf{R}_\theta^{i_1} & \cdots & \mathbf{R}_\theta^{i_{\tau-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_\theta^{i_0(\tau-1)} & \mathbf{R}_\theta^{i_1(\tau-1)} & \cdots & \mathbf{R}_\theta^{i_{\tau-1}(\tau-1)} \end{bmatrix}.$$

As in the proof of Theorem 2 we can equivalently consider the system of equations

$$\mathbf{m}\tilde{\mathbf{G}}' = \mathbf{c},$$

where

$$\begin{aligned} \mathbf{m} &= [\mathbf{m}_0, \ldots, \mathbf{m}_{\tau-1}] = [\mathbf{m}_{0,0}, \mathbf{m}_{0,1}, \ldots, \mathbf{m}_{\tau-1,0}, \mathbf{m}_{\tau-1,1}] \text{ and} \\ \mathbf{c} &= [\mathbf{c}_{i_0}, \ldots, \mathbf{c}_{i_{\tau-1}}] = [\mathbf{c}_{i_0,0}, \mathbf{c}_{i_0,1} \ldots, \mathbf{c}_{i_{\tau-1},0}, \mathbf{c}_{i_{\tau-1},1}]. \end{aligned}$$

We need to analyze $\kappa(\tilde{\mathbf{G}}')$. Towards this end, using the eigenvalue decomposition of $\mathbf{R}_\theta$, we have

$$\tilde{\mathbf{G}}' = \begin{bmatrix} \mathbf{Q} & & \\ & \ddots & \\ & & \mathbf{Q} \end{bmatrix} \tilde{\mathbf{\Lambda}} \begin{bmatrix} \mathbf{Q}^* & & \\ & \ddots & \\ & & \mathbf{Q}^* \end{bmatrix}, \text{ where} \tag{11}$$

$$\tilde{\mathbf{\Lambda}} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \Lambda^{i_0} & \Lambda^{i_1} & \cdots & \Lambda^{i_{\tau-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda^{i_0(\tau-1)} & \Lambda^{i_1(\tau-1)} & \cdots & \Lambda^{i_{\tau-1}(\tau-1)} \end{bmatrix} \tag{12}$$

and Λ is specified in (2). Note that the pre- and post-multiplying matrices in (11) above are both unitary. Therefore $\kappa(\tilde{\mathbf{G}}')$ is the same as $\kappa(\tilde{\mathbf{\Lambda}})$.

Using Claim 2, we can permute the rows and columns of $\tilde{\mathbf{\Lambda}}$ to put it in block-diagonal form so that

$$\tilde{\mathbf{\Lambda}}_d = \begin{bmatrix} \tilde{\mathbf{\Lambda}}_d[0] & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{\Lambda}}_d[1] \end{bmatrix}$$

where $\tilde{\mathbf{\Lambda}}_d[0]$ and $\tilde{\mathbf{\Lambda}}_d[1]$ are a Vandermonde matrices with parameter sets $\{e^{\mathrm{i}\theta i_0}, \ldots, e^{\mathrm{i}\theta i_{\tau-1}}\}$ and $\{e^{-\mathrm{i}\theta i_0}, \ldots, e^{-\mathrm{i}\theta i_{\tau-1}}\}$ respectively. Using Theorem 1, the result follows. □

Remark 2. Both circulation permutation matrices and rotation matrices allow us to achieve a specified threshold for distributed matrix vector multiplication. The required storage

fraction γ_A is slightly higher for the circulant permutation case and it requires q to be prime. However, it allows for an efficient FFT based decoding algorithm. On the other hand, the rotation matrix case requires a smaller Δ_A, but the decoding requires solving the corresponding system of equations the complexity of which can be cubic in Δ_A. We note that when the matrix sizes are large, the decoding time will be negligible as compared to the worker node computation time; we discuss this in Section 6. In Section 6, we show results that demonstrate that the normalized mean-square error when circulant permutation matrices are used is lower than the rotation matrix case.

5 Distributed Matrix-Matrix Multiplication

Let $\theta = 2\pi/q$, where $q \geq n$ (n is the number of worker nodes) is an odd integer and let $\mathbf{R}_\theta$ denote the corresponding rotation matrix. We partition $\mathbf{A}$ into $\Delta_A = 2u_A$ block columns indexed as $\mathbf{A}_{0,0}, \mathbf{A}_{0,1}, \ldots, \mathbf{A}_{u_A-1,0}, \mathbf{A}_{u_A-1,1}$ with a similar decomposition and indexing for $\mathbf{B}$ into $2u_B$ block columns, such that $u_A u_B \leq n$. Consider the following choices for the encoding matrices $\mathbf{G}_A$ and $\mathbf{G}_B$.

$$\mathbf{G}_A = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \ldots & \mathbf{I} \\ \mathbf{I} & \mathbf{R}_\theta & \mathbf{R}_\theta^2 & \ldots & \mathbf{R}_\theta^{n-1} \\ \mathbf{I} & \mathbf{R}_\theta^2 & \mathbf{R}_\theta^4 & \ldots & \mathbf{R}_\theta^{2(n-1)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \mathbf{I} & \mathbf{R}_\theta^{u_A-1} & \mathbf{R}_\theta^{2(u_A-1)} & \ldots & \mathbf{R}_\theta^{(n-1)(u_A-1)} \end{bmatrix}, \text{ and}$$

$$\mathbf{G}_B = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \ldots & \mathbf{I} \\ \mathbf{I} & \mathbf{R}_\theta^{u_A} & \mathbf{R}_\theta^{2u_A} & \ldots & \mathbf{R}_\theta^{(n-1)u_A} \\ \mathbf{I} & \mathbf{R}_\theta^{2u_A} & \mathbf{R}_\theta^{4u_A} & \ldots & \mathbf{R}_\theta^{2(n-1)u_A} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \mathbf{I} & \mathbf{R}_\theta^{(u_B-1)u_A} & \mathbf{R}_\theta^{2(u_B-1)u_A} & \ldots & \mathbf{R}_\theta^{(n-1)(u_B-1)u_A} \end{bmatrix}.$$

The master node operates according to the encoding rule discussed previously (*cf.* (9)) for both $\mathbf{A}$ and $\mathbf{B}$. Thus, each worker node stores $\gamma_A = 1/u_A$ and $\gamma_B = 1/u_B$ fraction of $\mathbf{A}$ and $\mathbf{B}$ respectively. The i-th worker node computes the pairwise product of the matrices $\hat{\mathbf{A}}_{i,k}^T \hat{\mathbf{B}}_{i,l}$ for $k, l = 0, 1$ and returns the result to the master node. The master node needs to recover all pair-wise products of the form $\mathbf{A}_{i,\alpha}^T \mathbf{B}_{j,\beta}$ for $i \in [u_A], j \in [u_B]$ and $\alpha, \beta = 0, 1$. Let $\mathbf{Z}$ denote a $1 \times 4u_A u_B$ block matrix that contains each of these pair-wise products.

Theorem 4. The threshold for the rotation matrix based matrix-matrix multiplication scheme is $u_A u_B$. The worst case condition number is bounded by $O(q^{q-u_A u_B+6})$.

Proof. Let $\tau = u_A u_B$ and suppose that the workers indexed by $i_0, \ldots, i_{\tau-1}$ complete their tasks. Let $\mathbf{G}_{A,\ell}$ denote the ℓ-th block column of $\mathbf{G}_A$ (with similar notation for $\mathbf{G}_B$). Note

that the ℓ-th worker node computes

$$\hat{\mathbf{A}}_{\ell,k_1}^T \hat{\mathbf{B}}_{\ell,k_2} = \left(\sum_{\alpha\in[u_A],\beta\in\{0,1\}} \mathbf{G}_A(2\alpha+\beta, 2\ell+k_1)\mathbf{A}_{\alpha,\beta}^T \right) \left(\sum_{\alpha\in[u_B],\beta\in\{0,1\}} \mathbf{G}_B(2\alpha+\beta, 2\ell+k_2)\mathbf{B}_{\alpha,\beta} \right)$$
$$\equiv \mathbf{Z}\cdot(\mathbf{G}_A(:,2\ell+k_1)\otimes\mathbf{G}_B(:,2\ell+k_2)),$$

using the properties of the Frolicker product. Based on this, it can be observed that the decodability of $\mathbf{Z}$ at the master node is equivalent to checking whether the following matrix is full-rank.

$$\tilde{\mathbf{G}} = [\mathbf{G}_{A,i_0}\otimes\mathbf{G}_{B,i_0}|\mathbf{G}_{A,i_1}\otimes\mathbf{G}_{B,i_1}|\dots|\mathbf{G}_{A,i_{\tau-1}}\otimes\mathbf{G}_{B,i_{\tau-1}}].$$

To analyze this matrix, consider the following decomposition of $\mathbf{G}_{A,\ell}\otimes\mathbf{G}_{B,\ell}$, for $\ell\in[n]$.

$$\begin{aligned}
\mathbf{G}_{A,\ell}\otimes\mathbf{G}_{B,\ell} &= \begin{bmatrix} \mathbf{I} \\ \mathbf{R}_\theta^{\ell} \\ \vdots \\ \mathbf{R}_\theta^{\ell(u_A-1)} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} \\ \mathbf{R}_\theta^{\ell u_A} \\ \vdots \\ \mathbf{R}_\theta^{\ell u_A(u_B-1)} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Q}\mathbf{Q}^* \\ \mathbf{Q}\Lambda^{\ell}\mathbf{Q}^* \\ \vdots \\ \mathbf{Q}\Lambda^{\ell(u_A-1)}\mathbf{Q}^* \end{bmatrix} \otimes \begin{bmatrix} \mathbf{Q}\mathbf{Q}^* \\ \mathbf{Q}\Lambda^{\ell u_A}\mathbf{Q}^* \\ \vdots \\ \mathbf{Q}\Lambda^{\ell u_A(u_B-1)}\mathbf{Q}^* \end{bmatrix} \\
&= (\mathbf{I}_{u_A}\otimes\mathbf{Q}) \begin{bmatrix} \mathbf{I} \\ \Lambda^{\ell} \\ \vdots \\ \Lambda^{\ell(u_A-1)} \end{bmatrix} \left[\mathbf{Q}^*\right] \otimes (\mathbf{I}_{u_B}\otimes\mathbf{Q}) \begin{bmatrix} \mathbf{I} \\ \Lambda^{\ell u_A} \\ \vdots \\ \Lambda^{\ell u_A(u_B-1)} \end{bmatrix} \left[\mathbf{Q}^*\right] \\
&= \underbrace{((\mathbf{I}_{u_A}\otimes\mathbf{Q})\otimes(\mathbf{I}_{u_B}\otimes\mathbf{Q}))}_{\tilde{\mathbf{Q}}_1} \underbrace{\left(\begin{bmatrix} \mathbf{I} \\ \Lambda^{\ell} \\ \vdots \\ \Lambda^{\ell(u_A-1)} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} \\ \Lambda^{\ell u_A} \\ \vdots \\ \Lambda^{\ell u_A(u_B-1)} \end{bmatrix} \right)}_{\mathbf{X}_\ell} \underbrace{\left(\left[\mathbf{Q}^*\right]^{\otimes 2} \right)}_{\tilde{\mathbf{Q}}_2},
\end{aligned}$$

where the last step follows by using the properties of Frolicker products. Using the above decomposition, we have

$$[\mathbf{G}_{A,i_0}\otimes\mathbf{G}_{B,i_0}|\mathbf{G}_{A,i_1}\otimes\mathbf{G}_{B,i_1}|\dots|\mathbf{G}_{A,i_{\tau-1}}\otimes\mathbf{G}_{B,i_{\tau-1}}] = \tilde{\mathbf{Q}}_1[\mathbf{X}_{i_0}|\mathbf{X}_{i_1}|\dots|\mathbf{X}_{i_{\tau-1}}] \begin{bmatrix} \tilde{\mathbf{Q}}_2 & 0 & \dots & 0 \\ 0 & \tilde{\mathbf{Q}}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{\mathbf{Q}}_2 \end{bmatrix}.$$

Table 1: Condition number comparison for matrix-vector multiplication system with $n = 31$.

Scheme	ℓ	τ	Avg. Cond. Num.	Max. Cond. Num.
Real Vandermonde-I	1/29	29	1.1×10^{13}	2.9×10^{13}
Complex Vandermonde-I	1/29	29	12	55
Circulant Permutation Matrix Embedding-I	1/28	29	12	55
Rotation Matrix Embedding-I	1/29	29	12	55
Real Vandermonde-II	1/28	28	4.9×10^{12}	2.3×10^{13}
Complex Vandermonde-II	1/28	28	27	404
Circulant Permutation Matrix Embedding-II	1/27	28	27	404
Rotation Matrix Embedding-II	1/28	28	27	404

Table 2: Average computation and decoding time comparison for matrix-vector $\mathbf{A}^T\mathbf{x}$ multiplication system with $n = 31$. $\mathbf{A}$ is of size 20000×10080, $\mathbf{x}$ is of length 20000

Scheme	ℓ	τ	Avg. Comp. Time(sec.)	Dec. Time(sec.)
Complex Vandermonde	1/28	28	0.51	0.02
Circulant Permutation Matrix Embedding-II	1/27	28	0.24	0.03
Rotation Matrix Embedding-II	1/28	28	0.24	0.007

We can conclude that the invertibility and the condition number of $\tilde{\mathbf{G}}$ only depends on $[\mathbf{X}_{i_0}|\mathbf{X}_{i_1}|\ldots|\mathbf{X}_{i_{\tau-1}}]$ as the matrices pre- and post- multiplying it are both unitary. The invertibility of $[\mathbf{X}_{i_0}|\mathbf{X}_{i_1}|\ldots|\mathbf{X}_{i_{\tau-1}}]$ follows from an application of Claim 3 in the Appendix. The proof of Claim 3 also shows that upon appropriate permutation, the matrix $[\mathbf{X}_{i_0}|\mathbf{X}_{i_1}|\ldots|\mathbf{X}_{i_{\tau-1}}]$ can be expressed as a block-diagonal matrix with four blocks each of size $\tau \times \tau$. Each of these blocks is a Vandermonde matrix with parameters from the set $\{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\}$. Therefore, an application of Theorem 1 implies that the worst case condition number is at most $O(q^{q-\tau+6})$. □

6 Comparisons and Experimental Results

In this section, we compare the performance of our scheme to prior works. When the number of worker nodes n is odd, we can pick $q = n$ for the rotation matrix embedding (for both matrix-vector and matrix-matrix cases). In this scenario we obtain a worst case condition number of $O(q^{q-\tau+6})$. As argued before, real Vandermonde matrices have condition numbers that are exponential in q, i.e., they are much higher. The work of [15] demonstrates an upper bound of $O(q^{2(q-\tau)})$ which grows much faster than our upper bound in the parameter $q - \tau$. In numerical experiments, our worst case condition numbers are much smaller than the work of [15]; we discuss this in detail below.

In Table 1, we compare the average and maximal condition number of different schemes for matrix-vector multiplication. The system under consideration has $n = 31$ worker nodes and a threshold specified by the third column. The "Real Vandermonde" scheme corresponds to the scheme of [1] with real evaluation points uniformly sampled from -1 to 1. The

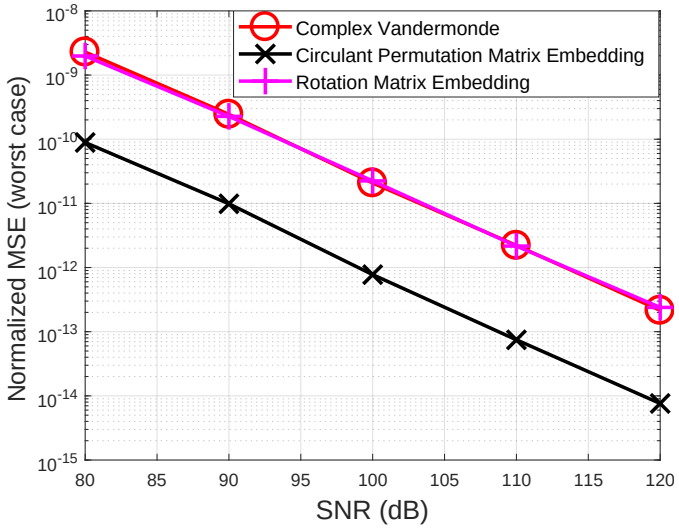

Figure 1: Consider matrix-vector $\mathbf{A}^T\mathbf{x}$ multiplication system with $n = 31$, $\tau = 28$. $\mathbf{A}$ has size 20000×10080 and $\mathbf{x}$ has length 20000.

Table 3: Condition number comparison for matrix-matrix multiplication system with $n = 31$, $u_A = 4$, $u_B = 7$

Scheme	Avg. Cond. Num.	Max. Cond. Num.
Real Vandermonde	4.9×10^{12}	2.3×10^{13}
Complex Vandermonde	27	404
Rotation Matrix Embedding	27	404
[15] scheme	1.2×10^{3}	6.4×10^{5}

"Complex Vandermonde" scheme chooses the evaluation points from 31-th roots of unity. We note here that the Complex Vandermonde scheme will incur higher computational cost at the worker nodes since it requires complex matrix multiplication. The "Circulant Matrix Embedding" and "Rotation Matrix Embedding" correspond to the schemes discussed in Sections 4. It can be observed from Table 1 that the both the worst case and the average condition numbers of our scheme are over eleven orders of magnitude better than the Real Vandermonde scheme.

Another point to be noted is the there is exact match of the condition number values for all the other schemes. This can be understood by following the discussion in Section 4. Specifically, our schemes have the property that the condition number only depends on the eigenvalues of corresponding circulation permutation matrix and rotation matrix respectively. These eigenvalues lie precisely in 31-th roots of unity.

For the matrix-matrix multiplication problem, the flop count for computing $\mathbf{A}^T\mathbf{B}$ is

Table 4: Average computation and decoding time comparison for matrix-vector $\mathbf{A}^T\mathbf{B}$ multiplication system with $n = 31$, $u_A = 4$, $u_B = 7$, $\mathbf{A}$ is of size 6000×3200, $\mathbf{B}$ is of 6000×4200.

Scheme	Avg. Comp. Time(sec.)	Dec. Time(sec.)
Complex Vandermonde	0.97	0.10
Rotation Matrix Embedding	0.35	0.04
[15] scheme	0.33	0.03

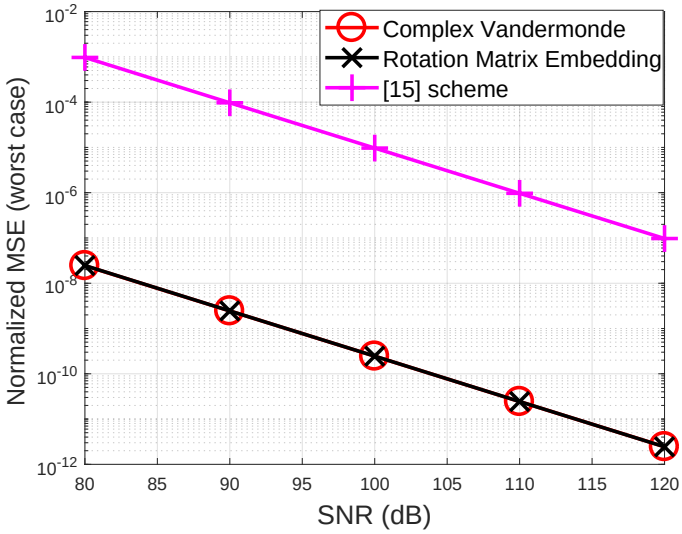

Figure 2: Consider matrix-matrix $\mathbf{A}^T\mathbf{B}$ multiplication system with $n = 31$, $\tau = 28$. $u_A = 4$, $u_B = 7$, $\mathbf{A}$ is of size 6000×3200, $\mathbf{B}$ is of 6000×4200.

$\approx 2rtw$. Therefore the flop count for the worker nodes is $\approx \frac{2rtw}{\tau}$. On the other hand, a simple calculation shows that the decoding flop count is independent of t for all the methods. Thus, when t is large, the decoding time will be negligible as compared to the worker node computation time. Nevertheless, from a practical perspective it is useful to understand the decoding times as well. Table 2 compares the average worker node computation time and the decoding time of the different schemes. The matrix $\mathbf{A}$ is of dimension 20000×10080 here. As expected the Complex Vandermonde scheme requires higher worker node computation time, whereas the Circulant Permutation and Rotation Matrix embeddings require almost the same time. The decoding time of the Rotation Matrix Embedding is lowest at 0.007 seconds. This is in spite of the fact that its decoding does not exploit any problem structure. The Circulant Permutation Matrix scheme requires decoding time of 0.03 seconds even though we can use FFT based approaches for it. We expect that for much larger scale problems, the FFT based approach may be faster, though this remains to be investigated.

We have not shown the results for the Real Vandermonde case here because the decoding failed to recover the correct answer in this case.

In Figure 1, we show the results of running experiments for comparing the normalized mean-squared error (MSE) of the different schemes. Let $\mathbf{A}^T\mathbf{x}$ denote the precise value of the computation and $\widehat{\mathbf{A}^T\mathbf{x}}$ denote the result of using one of the discussed methods. The normalized MSE is defined as $\frac{||\mathbf{A}^T\mathbf{x}-\widehat{\mathbf{A}^T\mathbf{x}}||_2}{||\mathbf{A}^T\mathbf{x}||_2}$. To simulate numerical precision problems, we added i.i.d. Gaussian noise (of different SNRs) to the result of the worker node computation. The master node then performs decoding on the noisy vectors. The plots in Figure 1 correspond to the worst case choice of worker nodes for each of the schemes. It can be observed that the Circulant Permutation Matrix Embedding has the best performance. This is because the many of the matrices on the block-diagonal in (8) have well-behaved condition numbers and only a few correspond to the worst case.

In the matrix-matrix multiplication scenario we again consider a system with $n = 31$ worker nodes and $u_A = 4$ and $u_B = 7$ so that the threshold $\tau = 28$. Once again we observe that the worst case condition number of the Rotation Matrix Embedding is about eleven orders of magnitude lower than the Real Vandermonde case. Furthermore, the scheme of [15] has a worst case condition number of 6.4×10^5 which is still over three orders of magnitude higher.

When the matrix $\mathbf{A}$ is of dimension 6000×3200 and $\mathbf{B}$ is of dimension 6000×4200, the worker node computation times and decoding times are listed in Table 4. As expected the Complex Vandermonde schemes take much longer for the worker node computations, whereas the Rotation Matrix Embedding and [15] take about the same time. The decoding times are also very similar. For the matrix-matrix case the normalized MSE is defined as $\frac{||\mathbf{A}^T\mathbf{B}-\widehat{\mathbf{A}^T\mathbf{B}}||_2}{||\mathbf{A}^T\mathbf{B}||_2}$ where $\mathbf{A}^T\mathbf{B}$ is the true product and $\widehat{\mathbf{A}^T\mathbf{B}}$ is the decoded product using one of the methods. As shown in Figure 2, the normalized MSE of our Rotation Matrix Embedding scheme is much about five orders of magnitude lower than the scheme of [15].

References

[1] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Proc. of Adv. in Neural Inf. Proc. Sys. (NIPS)*, 2017, pp. 4403–4413.

[2] L. Tang, K. Konstantinidis, and A. Ramamoorthy, "Erasure coding for distributed matrix multiplication for matrices with bounded entries," *IEEE Comm. Lett.*, vol. 23, no. 1, pp. 8–11, 2019.

[3] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *IEEE Int. Symp. on Inf. Theory*, 2017, pp. 2418–2422.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. on Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[5] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. of Adv. in Neural Inf. Proc. Sys. (NIPS)*, 2016, pp. 2100–2108.

[6] A. Mallick, M. Chaudhari, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," preprint, 2018, [Online] Available: https://arxiv.org/abs/1804.10331.

[7] S. Wang, J. Liu, and N. B. Shroff, "Coded sparse matrix multiplication," in *Proc. 35th Int. Conf. on Mach. Learning, ICML*, 2018, pp. 5139–5147.

[8] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *IEEE Int. Symp. on Inf. Theory*, 2018, pp. 1988–1992.

[9] A. B. Das, L. Tang, and A. Ramamoorthy, "C^{3}LES: Codes for coded computation that leverage stragglers," in *IEEE Inf. Th. Workshop*, 2018, pp. 1–5.

[10] V. Pan, "How Bad Are Vandermonde Matrices?" *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 2, pp. 676–694, 2016.

[11] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally Decodable Matrices for Distributed Matrix-Vector Multiplication," in *IEEE Int. Symp. on Inf. Theory*, 2019.

[12] A. B. Das, L. Tang, and A. Ramamoorthy, "C3LES: Codes for Coded Computation that Leverage Stragglers," in *IEEE Inf. Th. Workshop*, 2018.

[13] A. B. Das and A. Ramamoorthy, "Distributed Matrix-Vector Multiplication: A Convolutional Coding Approach," in *IEEE Int. Symp. on Inf. Theory*, 2019.

[14] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Random convolutional coding for robust and straggler resilient distributed matrix computation," 2019, [Online] Available: https://arxiv.org/abs/1907.08064.

[15] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," 2019, [Online] Available at: https://arxiv.org/abs/1903.08326.

[16] A. M. Subramaniam and A. Heidarzadeh and K. R. Narayanan, "Random Khatri-Rao-Product Codes for Numerically-Stable Distributed Matrix Multiplication," 2019, [Online] Available: https://arxiv.org/abs/1907.05965.

[17] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," 2018, [Online] Available: https://arxiv.org/abs/1801.07487.

[18] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the Optimal Recovery Threshold of Coded Matrix Multiplication," 2018, [Online] Available: https://arxiv.org/abs/1801.10292.

[19] N. J. Higham, *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics (SIAM), 2002 (2nd Ed.).

[20] Victor Y. Pan, "TR-2013003: Polynomial Evaluation and Interpolation: Fast and Stable Approximate Solution," 2013, [Online] Available: https://academicworks.cuny.edu/gc_cs_tr/378/.

A Vandermonde Matrix condition number analysis

Let $\mathbf{V}$ be a $m \times m$ Vandermonde matrix with parameters $s_0, s_1, \ldots s_{m-1}$. We are interested in upper bounding $\kappa(\mathbf{V})$. Let $s_+ = \max_{i=0}^{m-1} |s_i|$. Then it is known that $||\mathbf{V}|| \leq m \max(1, s_+^{n-1})$ [10]. Finding an upper bound on $||\mathbf{V}^{-1}||$ is more complicated and we discuss this in detail below. Towards this end we need the definition of a Cauchy matrix.

Definition 6. A $m \times m$ Cauchy matrix is specified by parameters $\mathbf{s} = [s_0 \ s_1 \ \ldots \ s_{m-1}]$ and $\mathbf{t} = [t_0 \ t_1 \ \ldots \ t_{m-1}]$, such that

$$\mathbf{C}_{\mathbf{s},\mathbf{t}} = \left(\frac{1}{s_i - t_j} \right)_{i,j=0}^{m-1}.$$

In what follows, we establish an upper bound on the condition number of Vandermonde matrices with parameters on the unit circle.

Theorem 5. Consider a $m \times m$ Vandermonde matrix $\mathbf{V}$ where $m < q$ (q is odd) with distinct parameters $\{s_0, s_1, \ldots, s_{m-1}\} \subset \{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\}$. Then,

$$\kappa(\mathbf{V}) \leq O(q^{q-m+6})$$

Proof. Recall that $\omega_q = e^{\mathrm{i}\frac{2\pi}{\mathrm{q}}}$ and $\omega_m = e^{\mathrm{i}\frac{2\pi}{\mathrm{m}}}$ and define $t_j = f\omega_m^j, j = 0, \ldots, m-1$ where f is a complex number with $|f| = 1$. We let $\mathbf{C}_{\mathbf{s},f}$ denote the Cauchy matrix with parameters $\{s_0, \ldots, s_{m-1}\}$ and $\{t_0, \ldots, t_{m-1}\}$. Let $\mathbf{W}$ be the m-point DFT matrix. The work of [10] shows that

$$\mathbf{V}^{-1} = \mathrm{diag}(f^{m-1-j})_{j=0}^{m-1} \mathbf{W}^* \mathrm{diag}(\omega_m^{-j})_{j=0}^{m-1} \mathbf{C}_{\mathbf{s},f}^{-1} \mathrm{diag}\left(\frac{1}{s_j^m - f^m} \right)_{j=0}^{m-1}.$$

It can be seen that the matrix $\text{diag}(f^{m-1-j})_{j=0}^{m-1}\mathbf{W}^*\text{diag}(\omega_m^{-j})_{j=0}^{m-1}$ is unitary. Therefore,

$$\begin{aligned}
||\mathbf{V}^{-1}|| &= ||\mathbf{C}_{\mathbf{s},f}^{-1}\text{diag}\left(\frac{1}{s_j^m - f^m}\right)_{j=0}^{m-1}|| \\
&\leq ||\mathbf{C}_{\mathbf{s},f}^{-1}|| \times \left(\frac{1}{\min_{i=0}^{m-1}|s_i^m - f^m|}\right) \\
&\leq m \times (\max_{i',j'}|(\mathbf{C}_{\mathbf{s},f}^{-1})_{i',j'}|) \times \left(\frac{1}{\min_{i=0}^{m-1}|s_i^m - f^m|}\right), \qquad (13)
\end{aligned}$$

where the last inequality comes from the inequality $||\mathbf{A}|| \leq ||\mathbf{A}||_F$.

In what follows, we upper bound the RHS of (13). Let $s(x)$ denote a function of x so that $s(x) = \Pi_{i=0}^{m-1}(x - s_i)$. The (i', j')-the entry of $\mathbf{C}_{\mathbf{s},f}^{-1}$ can be expressed as

$$\begin{aligned}
(\mathbf{C}_{\mathbf{s},f}^{-1})_{i',j'} &= (-1)^m s(t_{j'})(s_{i'}^m - f^m)/(s_{i'} - t_{j'}), \text{ so that} \\
|(\mathbf{C}_{\mathbf{s},f}^{-1})_{i',j'}| &= |s(t_{j'})||s_{i'}^m - f^m|/|s_{i'} - t_{j'}| \\
&\leq |s(t_{j'})|(|s_{i'}^m| + |f^m|)/|s_{i'} - t_{j'}| \\
&= 2|s(t_{j'})|/|s_{i'} - t_{j'}|.
\end{aligned}$$

Let $\mathcal{M} = \{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\} \setminus \{s_0, s_1, \ldots, s_{m-1}\}$ denote the q-th roots of unity that are *not* parameters of $\mathbf{V}$. Note that

$$\begin{aligned}
s(t_{j'}) &= \Pi_{i=0}^{m-1}(t_{j'} - s_i) \\
&= \left.\frac{x^q - 1}{\Pi_{\alpha_j \in \mathcal{M}}(x - \alpha_j)}\right|_{x=t_{j'}}, \text{ so that} \\
|s(t_{j'})| &= \frac{|t_{j'}^q - 1|}{\Pi_{\alpha_j \in \mathcal{M}}|t_{j'} - \alpha_j|} \\
&\leq \frac{2}{\Pi_{\alpha_j \in \mathcal{M}}|t_{j'} - \alpha_j|}.
\end{aligned}$$

Thus, we can conclude that

$$\begin{aligned}
|(\mathbf{C}_{\mathbf{s},f}^{-1})_{i',j'}| &\leq 4 \max_{i',j'} \frac{1}{\Pi_{\alpha_j \in \mathcal{M}}|(t_{j'} - \alpha_j)|} \frac{1}{|s_{i'} - t_{j'}|} \qquad (14) \\
&= 4\left(\frac{1}{\min_{i',j'}\Pi_{\alpha_j \in \mathcal{M}}|(t_{j'} - \alpha_j)|} \frac{1}{|s_{i'} - t_{j'}|}\right) \qquad (15)
\end{aligned}$$

Note that in the expression above, $s_{i'}$ is a parameter of $\mathbf{V}$ while the α_j's are the points within $\Omega_q = \{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\}$ that are not parameters of $\mathbf{V}$. Next, we choose $f = e^{\mathrm{i}\frac{\pi}{m}}$ so that $t_{j'} = f\omega_m^{j'} = e^{\mathrm{i}\pi/m}\omega_m^{j'}$. Next, we determine an upper bound on the RHS of (15).

Towards this end, we note that the distance between two points on the unit circle can be expressed as $2\sin(\theta/2)$ if θ is the induced angle between them. Furthermore, we have $2\sin(\theta/2) \geq 2\theta/\pi$ as long as $\theta \leq \pi$.
It can be seen that the closest point to $t_{j'}$ that lies within Ω_q has an induced angle

$$\frac{2\pi\ell}{q} - \frac{2\pi(j' + \frac{1}{2})}{m} \geq \frac{2\pi}{qm}\frac{1}{2} \geq \frac{\pi}{q^2}.$$

Therefore, the corresponding distance is lower bounded by $2/q^2$. Similarly, the next closest distance is lower bounded by $2/q$, followed by $2(2/q), 3(2/q), \ldots, (q-m-1)(2/q)$. Let $d = q - m$, Then,

$$\begin{aligned}
&\left(\Pi_{\alpha_j \in \mathcal{M}} |(t_{j'} - \alpha_j)|\right) \min_{i',j'} |s_{i'} - t_{j'}| \\
&\geq 2/q^2 \times 2/q \times 4/q \times \cdots \times 2(d-1)/q \times 2/q^2 \\
&= 2^{d+1}(d-1)!\frac{1}{q^{d+3}}.
\end{aligned}$$

Therefore,

$$|(\mathbf{C}_{\mathbf{s},f}^{-1})_{i',j'}| \leq \frac{q^{d+3}}{C_d}$$

where $C_d = 2^{d+1}(d-1)!$ is a constant. Let the i-th parameter $s_i = e^{\mathrm{i}2\pi\ell/\mathrm{q}}$. Then,

$$\begin{aligned}
|s_i^m - f^m| &= |e^{\mathrm{i}2\pi\ell\mathrm{m}/\mathrm{q}} + 1| \\
&= 2|\cos(\pi\ell m/q)|.
\end{aligned}$$

The term ℓm can be expressed as $\ell m = \beta q + \eta$ for integers β and η such that $0 \leq \eta \leq q-1$. Now note that $\eta \neq q/2$ since by assumption q is odd. Thus, $|\cos(\pi\ell m/q)|$ takes its smallest value when $\eta = (q+1)/2$ or $(q-1)/2$. In this case

$$\begin{aligned}
|\cos(\pi\ell n/q)| &= \left|\cos\left(\beta\pi + \pi\frac{q+1}{2q}\right)\right| \\
&\geq \left|\sin\left(\frac{\pi}{2q}\right)\right| \\
&\geq \frac{1}{q}.
\end{aligned}$$

Thus, we can upper bound the RHS of (13) and obtain

$$\begin{aligned}
||\mathbf{V}^{-1}|| &\leq m\frac{q^{d+3}}{C_d}q \\
&\leq \frac{q^{d+5}}{C_d}.
\end{aligned}$$

Finally, using the fact that $||V|| \leq m < q$. we obtain

$$\kappa(\mathbf{V}) \leq \frac{q^{d+6}}{C_d}.$$

□

Claim 2. Let $\mathbf{M}$ be a $l_1 q \times l_2 q$ matrix consisting of blocks of size $q \times q$ denoted by $\mathbf{M}_{i,j}$ for $i \in [l_1], j \in [l_2]$. Each $\mathbf{M}_{i,j}$ is a diagonal matrix. Then, the rows and columns of $\mathbf{M}$ can be permuted to obtain $\mathbf{M}^\pi$ which is a block diagonal matrix where each block matrix is of size $l_1 \times l_2$ and there are q of them.

Proof. For an integer a, let $(a)_q$ denote $a \mod q$. In what follows, we establish two permutations

$$\pi_{l_1}(i) = l_1(i)_q + \lfloor i/q \rfloor, 0 \leq i < l_1 q$$
$$\pi_{l_2}(j) = l_2(j)_q + \lfloor j/q \rfloor, 0 \leq j < l_2 q$$

and show that applying row-permutation π_{l_1} and column-permutation π_{l_2} to $\mathbf{M}$ will result in a block diagonal matrix $\mathbf{M}^\pi$.

We observe that (i,j)-th entry in $\mathbf{M}$ is the $((i)_q, (j)_q)$-th entry in $\mathbf{M}_{\lfloor i/q \rfloor, \lfloor j/q \rfloor}$. Under the applied permutations the (i,j)-th entry in $\mathbf{M}$ is mapped to $(l_1(i)_q + \lfloor i/q \rfloor, l_2(j)_q + \lfloor j/q \rfloor)$-entry in $\mathbf{M}^\pi$. Recall that $\mathbf{M}_{\lfloor i/q \rfloor, \lfloor j/q \rfloor}$ is a diagonal matrix which implies that for $(i)_q \neq (j)_q$, the $(l_1(i)_q + \lfloor i/q \rfloor, l_2(j)_q + \lfloor j/q \rfloor)$ entry in $\mathbf{M}^\pi$ is 0. Therefore $\mathbf{M}^\pi$ is a block diagonal matrix with q blocks of size $l_1 \times l_2$. □

Claim 3. Let $a_0(z) = \sum_{j=0}^{\ell_a - 1} a_{j0} z^j$, $a_1(z) = \sum_{j=0}^{\ell_a - 1} a_{j1} z^{-j}$ and $b_0(z) = \sum_{j=0}^{\ell_b - 1} b_{j0} z^{j\ell_a}$, $b_1(z) = \sum_{j=0}^{\ell_b - 1} b_{j1} z^{-j\ell_a}$. Then, $a_{k_1}(z) b_{k_2}(z)$ for $k_1, k_2 = 0, 1$ are polynomials that can be recovered from $\ell_a \ell_b$ distinct evaluation points in $\mathbb{C}$.

Let $\mathbf{D}(z^j) = \text{diag}([z^j \ z^{-j}])$ and let

$$\mathbf{X}(z) = \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{D}(z) \\ \vdots \\ \mathbf{D}(z^{\ell_a - 1}) \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{D}(z^{\ell_a}) \\ \vdots \\ \mathbf{D}(z^{\ell_a(\ell_b - 1)}) \end{bmatrix}$$

Then, if z_i's are distinct points in $\mathbb{C}$, the matrix

$$[\mathbf{X}(z_1) | \mathbf{X}(z_2) | \ldots | \mathbf{X}(z_{\ell_a \ell_b})],$$

is nonsingular.

Proof. Firstly we show that $a_{k_1}(z)b_{k_2}(z)$ for $k_1, k_2 = 0, 1$ are polynomials that can be recovered from $\ell_a\ell_b$ distinct evaluation points in $\mathbb{C}$. Towards this end, these four polynomials can be written as

$$a_0(z)b_0(z) = \sum_{i=0}^{\ell_a-1}\sum_{j=0}^{\ell_b-1} a_{i0}b_{j0}z^{i+j\ell_a},$$
$$a_0(z)b_1(z) = \sum_{i=0}^{\ell_a-1}\sum_{j=0}^{\ell_b-1} a_{i0}b_{j1}z^{i-j\ell_a},$$
$$a_1(z)b_0(z) = \sum_{i=0}^{\ell_a-1}\sum_{j=0}^{\ell_b-1} a_{i1}b_{j0}z^{-i+j\ell_a}, \text{ and}$$
$$a_1(z)b_1(z) = \sum_{i=0}^{\ell_a-1}\sum_{j=0}^{\ell_b-1} a_{i1}b_{j1}z^{-i-j\ell_a}.$$

Upon inspection, it can be seen that each of the polynomials above has $\ell_a\ell_b$ consecutive powers of z. Therefore, each of these can be interpolated from $\ell_a\ell_b$ non-zero distinct evaluation points in $\mathbb{C}$.

The second part of the claim follows from the above discussion. To see this we note that

$$[a_0(z)\ a_1(z)] = [a_{00}\ a_{01}\ a_{10}\ a_{11}\ \dots a_{(\ell_a-1)0}\ a_{(\ell_a-1)1}] \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{D}(z) \\ \vdots \\ \mathbf{D}(z^{\ell_a-1}) \end{bmatrix} \text{ and}$$
$$[b_0(z)\ b_1(z)] = [b_{00}\ b_{01}\ b_{10}\ b_{11}\ \dots b_{(\ell_b-1)0}\ b_{(\ell_b-1)1}] \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{D}(z^{\ell_a}) \\ \vdots \\ \mathbf{D}(z^{\ell_a(\ell_b-1)}) \end{bmatrix}.$$

Furthermore, the four product polynomials under consideration can be expressed as

$$[a_0(z)\ a_1(z)] \otimes [b_0(z)\ b_1(z)]$$
$$= \left([a_{00}\ a_{01}\ a_{10}\ a_{11}\ \dots a_{(\ell_a-1)0}\ a_{(\ell_a-1)1}] \otimes [b_{00}\ b_{01}\ b_{10}\ b_{11}\ \dots b_{(\ell_b-1)0}\ b_{(\ell_b-1)1}]\right) \mathbf{X}(z).$$

We have previously shown that all polynomials in $[a_0(z)\ a_1(z)] \otimes [b_0(z)\ b_1(z)]$ can be interpolated by obtaining their values on $\ell_a\ell_b$ non-zero distinct evaluation points. This implies that we can equivalently obtain

$$\left([a_{00}\ a_{01}\ a_{10}\ a_{11}\ \dots a_{(\ell_a-1)0}\ a_{(\ell_a-1)1}] \otimes [b_{00}\ b_{01}\ b_{10}\ b_{11}\ \dots b_{(\ell_b-1)0}\ b_{(\ell_b-1)1}]\right)$$

which means that $[\mathbf{X}(z_1)|\mathbf{X}(z_2)|\dots|\mathbf{X}(z_{\ell_a\ell_b})]$ is non-singular.

□